

Week 2: Implementation of a classifier using an open source dataset

7/8/25

Aim: to implement a classifier using an open source data set specifically to classify the iris dataset using support vector machine (SVM)

Algorithm:

- 1, import necessary libraries like sklearn, matplotlib etc
- 2, load the dataset: use the iris dataset from sklearn dataset
- 3, preprocess the data:
 - * select first two features (for 2D visualisation)
 - * split the dataset into training and testing sets
 - * standardise the features using standardScaler
- 4, train the SVM model:
 - * initialise SVM with a linear kernel
 - * fit the model using training data
- 5, visualize the decision boundaries:
 - * create a meshgrid of input size / space
 - * predict the class for each point in the grid
 - * plot the regions along with actual training points.

output

accuracy: 60

confusion matrix:

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 13 \end{bmatrix}$$

no. of rows = 30
no. of columns = 3
dimensions of matrix = 3x3
and no. of rows = 30
no. of columns = 3

classification report

	Precision	recall	f1 score	support
0	1.00	1.00	1.00	19
1	1.00	1.00	1.00	13
2	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45

~~deed~~

observation:

steps:

1, load the iris dataset

2, split into training and test sets

3, train a classifier

4, evaluate the classifier

code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3,
    random_state=0, stratify=y)
```

sc = StandardScaler()

X_train_std = sc.fit_transform(X_train)

X_test_std = sc.transform(X_test)

```
SVM = SVC(Kernel='linear', C=1.0, random_state=0)
```

```
SVM.fit(X_train_std, y_train)
```

```
def plot_decision_regions(X, y, classifier, title):
    markers = ('s', 'x', 'o')
```

colors = ('red', 'blue', 'green')

cmap = plt.cm.RdYlBu

$x_1_{\min}, x_1_{\max} = X[:, 0].min() - 1, X[:, 0].max() + 1$

$x_2_{\min}, x_2_{\max} = X[:, 1].min() - 1, X[:, 1].max() + 1$

$x_1, x_2 = \text{np.meshgrid}(\text{np.arange}(x_1_{\min}, x_1_{\max}, 0.02), \text{np.arange}(x_2_{\min}, x_2_{\max}, 0.02))$

~~$z = \text{classifier.predict}(\text{np.array}([x_1.ravel(), x_2.ravel()])^T)$~~

$z = z.reshape(x_1.shape)$

```
plt.contourf(x1, x2, z, alpha=0.3, cmap=cmap)
```

```
plt.xlim(x1.min(), x1.max())
```

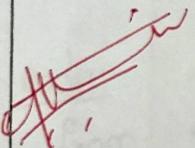
```
plt.ylim(x2.min(), x2.max())
```

Observations

- * Random forest classifier achieves high accuracy (97-100%) on the iris dataset, indicating the model effectively handles the dataset.
- * The dataset is relatively simple and well-separated.
- * So, classifier is reliable across categories.
- * This dataset is small and clean; performance might differ on more complex or noisy datasets.

```
for idx, cl in enumerate(np.unique(y)):
    plt.scatter(x=x[y == cl, 0], y=x[y == cl, 1],
                alpha=0.8, c=colors[idx],
                marker=markers[idx], label=''
                + iris.target_names[cl])
plt.xlabel('sepal length (standardised)')
plt.ylabel('sepal width (standardised)')
plt.title(title)
plt.legend()
plt.grid()
plt.figure(figsize=(7, 5))
plot_decision_region(x_train_std, y_train,
                      classifier=svm, title='SVM classifier')
plt.show()
```

Result: the SVM classifier was successfully implemented on the Iris dataset using two features.



```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n⌚ Classification Report:\n", classification_report(y_test, y_pred, target_names=iris.target_names))
print("\n📊 Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

✓ Accuracy: 1.0

⌚ Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	10
versicolor	1.00	1.00	1.00	9
virginica	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

📊 Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```